Mapping systolic FIR filter banks onto fixed-size linear processor arrays

Petkov, Nikolay

Published in:
EPRINTS-BOOK-TITLE

Publication date:
1990

MAPPING SYSTOLIC FIR FILTER BANKS ONTO FIXED-SIZE LINEAR PROCESSOR ARRAYS

Nikolay Petkov

University of Erlangen-Nürnberg
Institute of Informatics IMMD (III)
Martensstraße 3, 8520 Erlangen, West Germany

Abstract: A technique for mapping systolic FIR filter banks onto fixed-size processor arrays is presented. It is based on the time-sharing properties of *c*-slow circuits. The technique can be further developed to a formalism and holds high potential for automatic realization. It has been applied to the mapping of systolic filter banks onto a fixed-size array of Transputers.

Key words: FIR filters, convolution, systolic algorithms, *c*-slow circuits, Transputers,

Introduction

Since the explicit formulation of the notion *'systolic array'* [1-6], a great number of systolic algorithms has been proposed for different computational problems (The reader is referred to the monograph under [7] for an extensive introduction to systolic algorithms and arrays, and for many examples). In particular, the convolution and its major application in finite impulse response (FIR) filtering have been among the problems most exhaustively studied. Numerous systolic algorithms have been given for these problems operating both on the word and on the bit level. Since we are primarily concerned with the implementation of systolic algorithms on (appropriate) parallel computers, we focus here on the word-level algorithms [4-18].

One of the major problems in implementing systolic algorithms is that they usually require a processor array whose size depends on the size of the problem to be solved. For instance, the convolution of an input digital signal with a set of N coefficients would typically require a systolic array of $N/2$ or N cells/processors. However, unless a special purpose multiprocessor system is built for a specific application, it should be considered as a mere coincidence, if the number of processors required by a systolic array model of the algorithm equals the number of processors of the parallel computer which is available for implementation. In most practical cases, the former number is (much) greater than the latter one. The problem becomes even more complex when a whole filter bank with many channels having coefficient sets of different size needs to be implemented in a processor array of fixed size and fixed topology.

In this paper, it is shown how systolic FIR filter banks of arbitrary size and structure can be efficiently implemented in a processor array of fixed size and fixed topology. The approach used is based on the theory of the so called *c*-slow circuits and gives a discipline for mapping of systolic FIR filter banks onto a linear processor array of fixed size [16,17]. Since this approach is based on a set of well-defined rules, it is very suitable for automatic realization. The paper is organized as follows: Elements of the theory of *c*-slow circuits are presented in the next section. Then the technique is illustrated on several examples of filter banks. Some characteristics of the technique are summarized in the last section.

Using C-slow Circuits

The mapping technique proposed here is actually a time-sharing method for the use of logic circuits. It will be illustrated on automata consisting of combinatorial circuits and registers (delay elements). The automata abstraction should, however, not be considered as a proposal for hardware implementation, but rather as a model of an algorithm. This art of algorithm representation is widely used in the literature and it is especially useful and comprehensive for the representation of parallel systolic algorithms [7]. We illustrate the technique on a simple example, since it might be more illuminating than giving a general theory.

Figure 1a shows an accumulator which consists of an adder and an accumulation register (shown by a small black bar). The circuit is capable of accumulating and outputing the running sum $y_i = \Sigma_{-\infty}^{i} x_k$ of an input digital signal x_k, k = ... -1, 0, 1, ... Now, we change the circuit in Figure 1a by replacing the register by two (in general by c) chained registers, Figure 1b. The new circuit can realize the same function as the original one, if the time periods between consecutive data units measured in the number of cycles of the clock which controls the registers are increased twice (in general c times), Figure 1b. Since the new circuit is two (generally c) times slower than the original one, it is called after [19] a 2-slow (generally c-slow) version of the original circuit. A 2-slow (in general c-slow) circuit can be used for the concurrent execution of two (generally of c) independent computational problems. Figure 1c gives an illustration: the operations $y_i = \Sigma_{-\infty}^{i} x_k$ and $y'_i = \Sigma_{-\infty}^{i} x'_k$ are executed during the clock cycles for which holds $t_{\text{mod } 2} = 0$ and $t_{\text{mod } 2} = 1$, respectively.

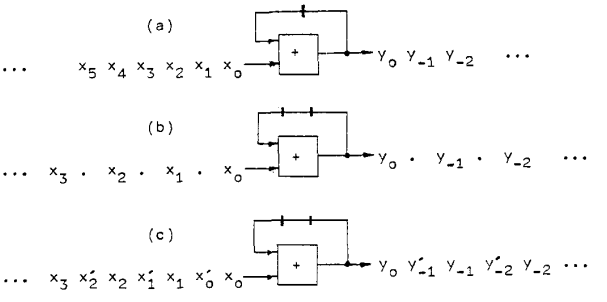


Figure 1 Construction and use of c-slow circuits (here $c = 2$)

Figure 2 shows how the method described above can be applied to array structures. The array shown in Figure 2a consists of three independent accumulators. These three parts of the array are active on three independent accumulation processes: $y_i = \Sigma_{-\infty}^{i} x_k$, $y'_i = \Sigma_{-\infty}^{i} x'_k$, and $y''_i = \Sigma_{-\infty}^{i} x''_k$, respectively. A 3-slow version of one accumulator can execute all three accumulation processes, Figure 2b. It is active on the first, second and third accumulation process in the clock cycles for which holds $t_{\text{mod } 3} = 0$, $t_{\text{mod } 3} = 1$, and $t_{\text{mod } 3} = 2$, respectively. In this way, a c-slow version of an appropriate $(1/c)$-th part of a homogeneous array can execute the task of the whole array. (In doing this, it is, however, c times slower than the original circuit.) The method can be generalized for the case in which the parts of the circuit are interconnected. To take account of the interconnections, feedbacks and multiplexers should be added to the model. For this case and for further details on the technique, the reader is referred to [17].

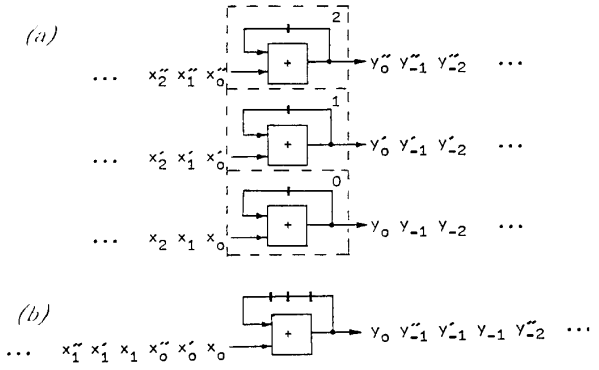
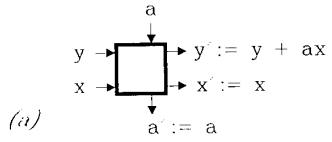


Figure 2 A c-slow version of a $(1/c)$-th part of a homogeneous array can do the work of the whole array (here $c = 3$)

Mapping Filter Banks onto a Linear Processor Array

For simplicity of representation, only small-size examples are considered in the following. The target system is assumed to have only three processor nodes connected into a a linear array. (The actual target system we use is an array of 10 Transputers.) Generalizations for an arbitrary number of processing nodes are straightforward and will not be considered here.



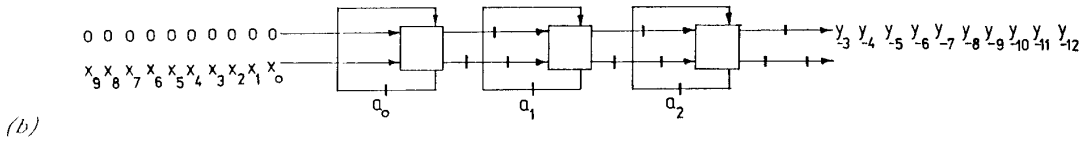


Figure 3 Systolic array for convolution with 3 coefficients

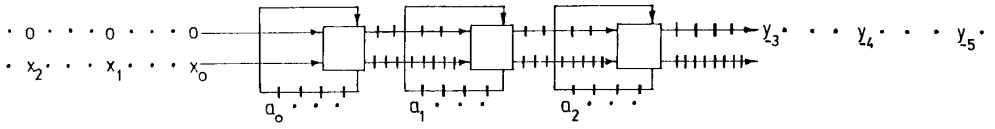


Figure 4 A 4-slow systolic convolver of three cells

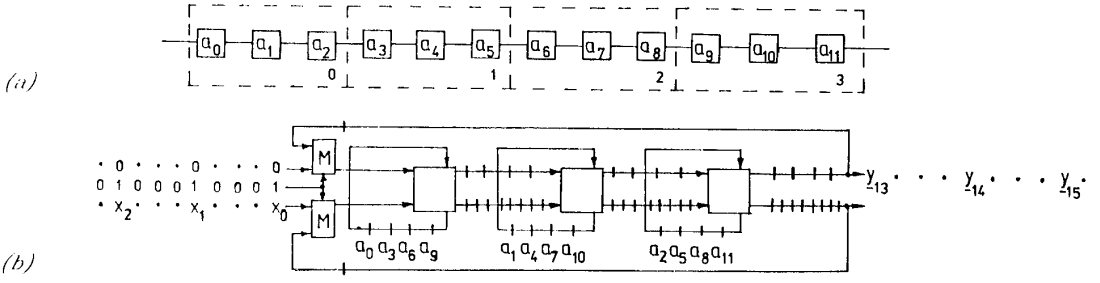


Figure 5 Using a 3-cell, 4-slow systolic array for a 12-coefficient convolver

Figure 3 shows a three-cell systolic array for implementing the convolution (FIR filter) $y_i = \Sigma_{j=0}^{N-1} a_j x_{i-j}$ of an input digital signal x_k, k = ... -1, 0, 1, ... , with a set $\{a_i | i = 0, 1, \ldots, N\text{-}1\}$ of three coefficients ($N = 3$). The (combinatorial) function of one cell is specified in Figure 3a. The implementation of this algorithm on the target (three processor) system is straightforward: cells are mapped onto processors and delay elements are implemented as locations in the private memory or local communication memory of the respective processing nodes. In the case of Transputer implementation, cells and delay elements are realized as sequentially replicated (OCCAM) processes which execute concurrently and communicate through channels.

Figure 4 shows a 4-slow version ($c = 4$) of the array shown in Figure 3. Each register of the array in Figure 3 has been replaced by a chain of four registers. The number of clock cycles between consecutive input/output operations is increased four times. The 4-slow systolic array shown in Figure 4 is capable of executing concurrently four different convolutions. It means that by time sharing, it can do the work of four systolic arrays like the one in Figure 3 (in doing this, it is, however, four times slower). This property will next be used for the realization of systolic filter banks.

The first example is a filter bank of just one channel/-filter. The number of coefficients of this filter is, however, greater than the number of processor nodes available in the target system, so that a one-to-one mapping of coefficients onto processors is not possible. Figure 5a shows a block diagram of a 12-cell systolic array for the convolution with 12 coefficients. The systolic array itself is assumed to be of the kind shown in Figure 3 and to consist of 12 cells, one cell for each coefficient. Such an array can be decomposed into four connected parts (subarrays), each of which is identical with the array shown in Figure 3. Figure 5b illustrates how a 4-slow version of one 3-cell array can do the work of the 12-cell array by executing the operations of subarrays 0, 1, 2, and 3 in the clock cycles for which holds

$t_{\text{mod }4} = 0$, $t_{\text{mod }4} = 1$, $t_{\text{mod }4} = 2$, and $t_{\text{mod }4} = 3$, respectively. The multiplexers in front of the array transfer input data when the control bit is 1. Otherwise, the feedbacks are enabled. In this way, a 12-coefficient systolic convolver is realized by a 3-cell systolic array. The transformed algorithm model uses only three cells and can thus be directly mapped onto an array of three processor nodes. (In practice, it is more convenient to use one more processor node which carries out the function of the multiplexers used in the model and which is also used as an interface to the host computer. Thus a ring configuration is actually used.)

Another example is shown in Figure 6. It is a filter bank with two channels, each of them with six coefficients. A straightforward implementation would require two systolic arrays, each of six cells. We can, however, decompose the original model arrays into four 3-cell subarrays which are schematically shown in Figure 6a. A 4-slow version of one 3-cell subarray can do the work of all four subarrays, Figure 6b. Thus two 6-coefficient systolic convolvers are realized with a single 3-cell systolic array. The transformed algorithm model shown in Figure 6b can now easily be mapped onto the target array of three processing nodes.

One more example is shown in Figure 7. Figure 7a shows schematically three systolic convolvers with six, three, and three coefficients, respectively. The whole system consists again of four 3-cell subarrays whose work can be done by a 4-slow version of one 3-cell subarray, Figure 7b. The clock cycles for which the condition $t_{\text{mod }4} = 0$ or $t_{\text{mod }4} = 1$ holds are used for the tasks of the 6-cell array, and the clock cycles for which $t_{\text{mod }4} = 2$ and $t_{\text{mod }4} = 3$ holds are used for the two 3-cell arrays, respectively.

In general, a c-slow version of an N-cell systolic convolution array can be used to concurrently execute the tasks of n systolic arrays (filter channels) with $p_1N, p_2N, \ldots, p_nN$ coefficients, respectively, where

$$p_1 + p_2 + \ldots + p_n = c \,. \tag{1}$$

More generally, if a systolic filter bank of n channels with numbers of coefficients $N_1, N_2, \ldots, N_n$, respectively, is to be realized by a target system of N processing nodes, a c-slow version of an N-cell systolic array can be used as an algorithm model, where the factor c is determined according to the following relation

$$c = \lceil N_1/N \rceil + \lceil N_2/N \rceil + \ldots + \lceil N_n/N \rceil \tag{2}$$

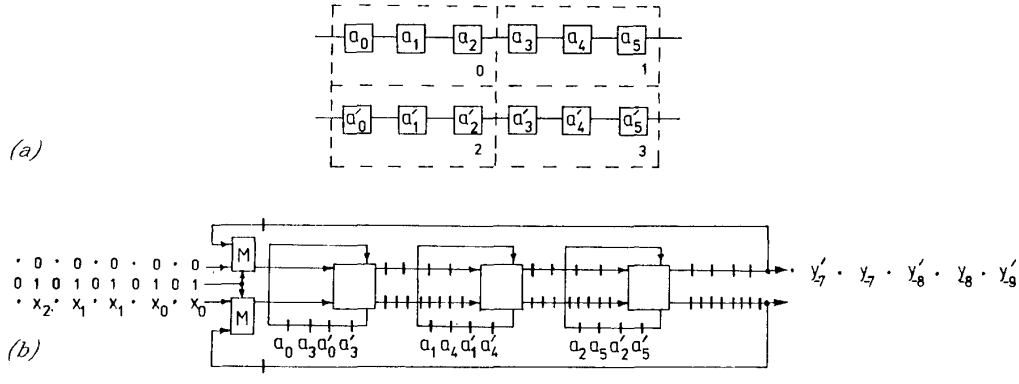


Figure 6 Using a 3-cell, 4-slow systolic array for two 6-coeffifficient convolvers

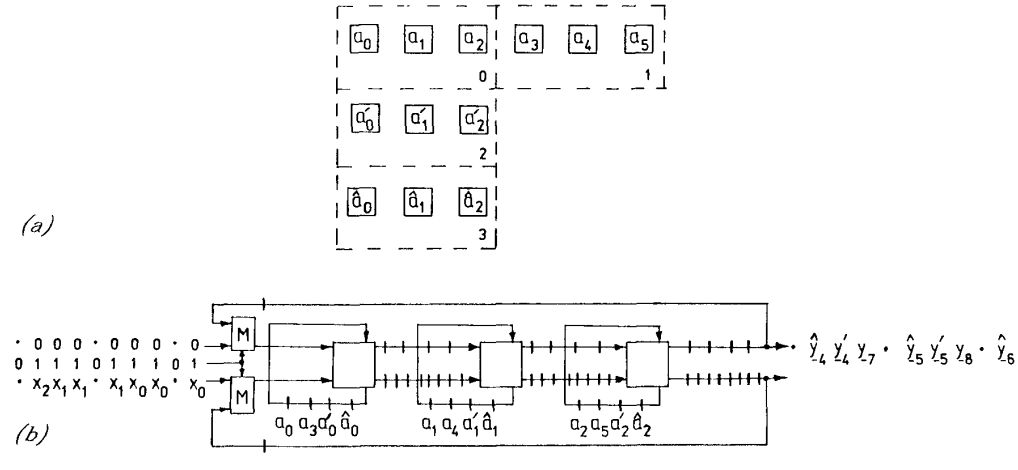


Figure 7 Using a 3-cell, 4-slow systolic array for one 6-coefficient and two 3-coefficient convolvers

Concluding Remarks

The major features of the mapping technique presented can be summarized as follows:

(i) The regularity of the algorithms is retained. The fixed-size systolic array to be used for a whole filter bank is as regular as the original problem-size dependent systolic array which realizes just one filter.

(ii) The additional structures represented in the model by feedbacks and multiplexers do not depend on the size of the filter bank to be implemented. The size of the bank and its particular structure are encoded in the control signals for the multiplexers.

(iii) The absence of transfer of intermediate results to the host is retained. Thus minimal communication with the host is guaranteed.

(iv) The whole mapping process is well-defined and can be carried out in a highly automatic fashion.

The mapping technique presented above can be used for any multiprocessor system of appropriate structure (ring or linear array with nearest neighborhood). The processor nodes to be used should, however, have enough private or communication memory in order to realize data structures as chains of delay elements which are required to implement c-slow versions. In some processor arrays, as for example the Carnegie-Mellon WARP machine, neighboring processors communicate via hardware-supported register chains of programmable length. The mapping technique proposed here is, therefore, especially suitable for such processor arrays. In other cases such as the Transputer array we use, these data structures should be organized in the local memory of each processor node.

Currently, a program system is being developed which automatically maps systolic FIR filter banks onto two fixed-size processor arrays: an array of Transputers and the DIRMU reconfigurable multiprocessor kit built at IMMD (III) in Erlangen.

The work presented in this paper focuses on the implementation of systolic algorithms in parallel computers. Note, however, that this mapping technique can successfully be used for the realization of flexible systolic filter banks in a VLSI chip. The systolic array shown in Figures 5, 6, and 7 can, for instance, be implemented in a VLSI chip. As shown by these figures, one and the same systolic array is used to realize different filter banks. The information on the kind of filter bank to be realized is encoded in the flow of control bits for the multiplexers.

Acknowledgements

The author gratefully appreciates the financial support of the Alexander von Humboldt Foundation, under whose research award this work has been carried out. The kind hospitality of the Institute for Mathematical Machines and Data Processing at the University of Erlangen-Nürnberg, in particular of its department (III) for computer structures, is also highly appreciated.

References

[1] H. T. Kung and C. E. Leiserson, "Systolic arrays (for VLSI)", *Sparse Matrix Proc.* 1978 (Society for Industrial and Applied Mathematics, 1979) pp. 256-282.

[2] H. T. Kung and C. E. Leiserson, "Algorithms for VLSI processor arrays", in *Introduction to VLSI Systems* by C. Mead and L. Conway (Reading: Addison-Wesley, 1980), Sec. 8.3

[3] H. T. Kung, "Let's design algorithms for VLSI systems", *Proc. Conf. VLSI: Architecture, Design, Fabrication,* Caltech, Jan. 1979, pp. 65-90

[4] H. T. Kung, "Why systolic architectures", *Computer,* vol. **15**, No. 1 (Jan. 1982), pp. 37-46

[5] H. T. Kung, "Special-purpose devices for signal and image processing: an opportunity in very large scale integration (VLSI)", *SPIE*, vol. **241** (1980) *Real-Time Signal Processing III*, pp.76-84

[6] H. T. Kung, "The structure of parallel algorithms", *Advances in Computers*, vol.**19** (New York: Academic, 1980),pp.65-111

[7] N. Petkov, *Systolische Algorithmen und Arrays* (Berlin: Akademie-Verlag, 1989, in German)

[8] N. Petkov-Turkedjiev, *Beitrag zur Theorie und Synthese systolischer Algorithmen und Arrays*, Dissertation B (habilitation), Technische Universität Dresden, 1987 (in German)

[9] H. T. Kung, L. M. Ruane, and D. W. L. Yen, "A two-level pipelined systolic array convolutions", in *VLSI Systems and Comptations*, edited by H. T. Kung et al. (Rockville, MD: Computer Science Press, 1981) pp.255-264

[10] J. A. E. Simons, "Systolic convolution array", *Electr.Letters* vol. **19**, No. 21 (1983), pp.874

[11] B. R. Mercy, "Systolic array technique applied to symmetric FIR filters", *Proc. ICASSP 1984*, pp.34.A.1.1-4

[12] O. Ersoy, "Semisystolic array implementation of circular, skew circular, and linear convolutions", *IEEE Trans. Computers*, vol. **C-34**, No. 2 (1985), pp.190-196

[13] N. Petkov-Turkedjiev, "Synthesis of systolic algorithms and processor arrays", *Lecture Notes on Computer Science,* vol. **237**: *Proc. CONPAR 86, Conf. Algorithms and Hardware for Parallel processing*, Aachen, Sept. 1986, eds. W. Händler et al. (Heidelberg: Springer-Verlag, 1986), pp.165-172

[14] N. Petkov and Fr. Sloboda, "Bit-level systolic array for digital curve smoothing", *Parallel Computing*, vol. **12**, No.3 (Dec. 1989), (Amsterdam: North-Holland, 1989) pp. 301-314

[15] N. Petkov, "A bidirectional systolic architecture with a single interface to the host", *Proc. of the Workshop on Languages and Systems for Parallel Processing*, Schmitten/-Arnoldshein, West Germany, January 26-26, 1990 (Darmstadt, West Germany: PARS Communications, 1990, in print)

[16] N. Petkov-Turkedjiev: "Systolische Filterbanken für Spracherkennung", *Proc. of 20. Fachkolloquium Informationstechnik*, TU Dresden, Febr. 1987, S.69-72

[17] N. Petkov, "Utilizing fixed-size systolic arrays for large computational processes", to be published in *Lecture Notes on Computer Science: "Recent Issues in Image Analysis and Processing"* (Heidelberg: Springer-Verlag, in print)

[18] H. Umeo "A design of time-optimum and register-number minimum systolic convolvers", *Parallel Computing*, vol. 12, No.3 (Dec. 1989) pp.285-300

[19] C. E. Leiserson, F. M. Rose, and J.B. Saxe, "Optimizing synchronous circuitry by retiming", *Proc. of the 3rd Caltech Conf. on VLSI*, (Rockville MD: Computer Science Press, 1983), pp. 87-116